Downloaded from SAE International, Friday, January 31, 2025

ARTICLE INFO
Article ID: 12-07-03-0020
© 2024 The Authors
doi:10.4271/12-07-03-0020

Employing a Model of Computation for Testing and Verifying the Security of Connected and Autonomous Vehicles

Ala Jamil Alnaser,[1] James Holland,[2] and Arman Sargolzaei[2]

[1]Florida Polytechnic University, Mathematics, USA
[2]University of South Florida, Mechanical Engineering Department, USA

Abstract

Testing and verifying the security of connected and autonomous vehicles (CAVs) under cyber-physical attacks is a critical challenge for ensuring their safety and reliability. Proposed in this article is a novel testing framework based on a model of computation that generates scenarios and attacks in a closed-loop manner, while measuring the safety of the unit under testing (UUT), using a verification vector. The framework was applied for testing the performance of two cooperative adaptive cruise control (CACC) controllers under false data injection (FDI) attacks. Serving as the baseline controller is one of a traditional design, while the proposed controller uses a resilient design that combines a model and learning-based algorithm to detect and mitigate FDI attacks in real-time. The simulation results show that the resilient controller outperforms the traditional controller in terms of maintaining a safe distance, staying below the speed limit, and the accuracy of the FDI estimation.

History

Received: 25 Aug 2023
Revised: 09 Jan 2024
Accepted: 15 Feb 2024
e-Available: 05 Mar 2024

Keywords

Connected and autonomous vehicles, Coverage, Model of computation, Safety and security, Testing and verification framework, Secure cooperative adaptive cruise control

Citation

Alnaser, A., Holland, J., and Sargolzaei, A., "Employing a Model of Computation for Testing and Verifying the Security of Connected and Autonomous Vehicles," *SAE Int. J. of CAV* 7(3):309–323, 2024, doi:10.4271/12-07-03-0020.

ISSN: 2574-0741
e-ISSN: 2574-075X

© 2024 The Authors. Published by SAE International. This Open Access article is published under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0/), which permits distribution, and reproduction in any medium, provided that the original author(s) and the source are credited.

Downloaded from SAE International, Friday, January 31, 2025

I. Introduction

Autonomous vehicles are one of the most promising solutions to the exceedingly many and complex transportation problems. Based on a report by the National Highway Traffic Safety Administration (NHTSA) [1], advanced driver assistance systems (ADAS) technologies have the potential to prevent about 62% of traffic fatalities. However, ADAS cannot function fully independently from the human driver, which is the source of the vast majority of incidents [2]. The natural evolution of ADAS technology will enable the jump to completely autonomous driving by progressively eliminating the vehicle's dependence on human drivers, which gives rise to the hybridized approach known as connected and autonomous vehicles (CAVs). CAVs have the potential to drastically reduce accidents, improve transportation system efficiency, reduce gas emissions, and much more. Recent rapid advancements in machine intelligence, machine vision, processing speed, and sensor fusion technology are enabling CAV technology to become readily available for widespread, everyday use in the very near future [3].

Despite all of the advantages of this technology, two significant barriers stand in the way of wide-scale adoption. The first is the lack of a testing and verification protocol that ensures operational safety and security. This challenge is further complicated by the fact that CAVs rely on various sensors, actuators, communication networks, and software components to operate autonomously and cooperatively [4, 5]. These components are subject to faults and attacks that can compromise the safety and security of CAVs and their passengers. Faults can occur due to hardware failures, software bugs, environmental disturbances, or human errors [6, 7, 8, 9]. Moreover, attacks can be launched by malicious actors who aim to disrupt, deceive, or damage CAVs and their infrastructure. Some common types of attacks include FDI, denial of service, replay, spoofing, and jamming [10, 11, 12, 13, 14, 15, 16, 17, 18]. These faults and attacks can affect the perception, planning, decision-making, and control of CAVs, leading to undesirable outcomes such as collisions, violations, delays, or loss of control. The second barrier to wide-scale adoption is that these vulnerabilities exponentially increase the complexity of testing [19, 20]. Therefore, it is essential to design and test CAVs with robust and resilient mechanisms that can detect and mitigate faults and attacks in real-time.

To address this, a process that builds an engineering argument for ensuring safety must be developed. Typically, this argument is built based on the following principles. First, a conceptual understanding of the problem is built and supported through virtual models. Second, a testing process is built to validate the model and build an argument for correctness. Third, the state space of tests is examined within the modeling environment to develop metrics for completeness. Finally, a structure is constructed where field testing feeds back into this flow, so safety is always rising and leading to accumulated learning [21].

In testing and verifying CAVs, there are various approaches to tackle this challenging endeavor [22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]. Simulated testing encompasses both offline and real-time methods [22]. Offline simulations emphasize optimizing computational speed, while real-time simulations prioritize testing accuracy within a constrained response time [23]. Additionally, real-time simulations guarantee access to testing data, expediting the validation phase and enhancing certainty throughout the development process [24]. The other existing verification solutions utilize ad hoc methods, such as miles driven, to demonstrate some indication of operational safety. This often assumes that the CAV's perception of the surrounding environment and the environment, itself, is comprehensive and perfect [36]. However, these approaches often lack the coverage and completeness required to test rare and extreme cases where the CAV under examination is prone to failure. In addition, the above-mentioned approaches are only focus on one aspect of CAVs such as testing the perception algorithms. To the best of our knowledge, no fundamental structure has been developed to demonstrate the security and reliability of CAV products, only operational success under ideal circumstances. This research seeks to address all the aforementioned shortcomings of the current state-of-the-art by providing a testing framework and mathematically sound metrics to quantify a CAV's security and reliability.

In this article, we present a framework for testing and verifying the decision-making capabilities of CAVs while the vehicle is under attacks. The framework is based on two mathematically defined procedures. The first is for generating scenarios while the second is for testing the CAV's responses. Our framework leverages an artificial neural network (ANN) to generate a seed for each scenario, which dictates the first scene in testing. Furthermore, we demonstrate the effectiveness of our algorithm as a testing framework by subjecting two different ADAS controllers to two, identical scenarios. The first controller is a traditionally designed model while the second is a resilient controller designed to detect and mitigate false data injection (FDI) attack. The first scenario that they were subjected to seeks to analyze performance while operating under ideal circumstances with the second scenario being designed to inject noise into the vehicle communication network. The purpose of this additional phase of research is to outline where our framework succeeds and understand where it fails such that we can confidently continue building this proof-of-concept into our vision of a comprehensive CAV testing framework.

In summary, the contributions of this work are as follows: (i) We proposed a unique algorithm to generate scenes in a closed-loop manner that allows for the identification of equivalent scenes. This, in turn, enables the production of unique testing scenarios tailored for the system being tested under attacks such as FDI. (ii) Since the verification function is incorporated within the scenario generation algorithm, the response of the controller is tested continuously and converges toward scenarios that maximize unsafe behavior by the CAV. (iii) We describe a secure cooperative adaptive cruise control (CACC) under FDI attacks where we implemented the proposed framework to verify its safety and security.

Downloaded from SAE International, Friday, January 31, 2025

The article is organized as follows: Section II presents a summary of current relevant results. Section III provides the necessary mathematical background for the proposed testing and verification framework. The framework is then discussed in Section IV. The novel resilient control algorithm utilized for CACC under testing is presented in Section V. To briefly describe the resilient controller, it combined a model and learning-based algorithms to detect and mitigate FDI attacks in real-time. Finally, the proposed framework is implemented to test the safety of the CACC algorithms under FDI attacks, where the results are discussed in Section VI.

II. Literature Review

In the past several years both the academic community and industry have focused on the testing and verification of CAVs. By leveraging classical methods, the field has developed novel approaches targeted at testing and verifying the readiness of CAVs. In [25] the authors identified five key testing challenges: driver-less scenarios, complex requirements, non-deterministic algorithms, inductive learning, and fail-operational systems. The authors proposed phased deployment, architecture changes, and fault injection as potential solutions, while suggesting a shift toward aligning existing software safety approaches with the vehicle's design process.

In [26] the authors presented an adaptive method to generate scenarios for accelerating the testing of general autonomous systems. This research sought to understand the decision-making processes of an autonomous system and identify the transient effects caused by transitioning between performance modes. However, their methodology treats the system as a black box, preventing the modification of the decision-making algorithms.

The work in [27] emphasized the importance of offline testing for validating autonomous vehicle performance and control algorithms across various virtual scenarios. It introduced a novel simulation platform with hardware in the loop (HIL), comprising four layers that simulated vehicle models, sensors, environments, and ECU control. This platform facilitated comprehensive closed-loop evaluations of perception, planning, decision-making, and control algorithms, enabling seamless migration to real self-driving cars. Experimental validations conducted in virtual scenarios of public roads and open parking lots substantiated the effectiveness of the simulation platform.

The authors in [28] employed procedural content generation and search-based testing to automatically create virtual scenarios, focused on testing the lane-departure software of CAVs. While [29] used a game theoretic traffic model to test the control of autonomous vehicles' decisions and to calibrate the parameters of an existing control system. In [30], several components, such as road geometry, the environment, and the behavior of the dynamic object, were formally defined and combined to generate a virtual reality. Additionally, Zofka et al. [31] included vehicle mechanics, sensors, and traffic to generate scenarios for testing simulations.

Leveraging a machine learning to generate test scenarios, the authors in [32] presented a system that used a learning feature utilizing feedback from the CAV's controller and aimed to converge on scenarios that model edge cases. Another approach that also used machine learning was presented in [33], employing recurrent neural networks (RNNs) that were applied to existing crash data to create test cases. Furthermore, the authors in [34] suggested an approach called "deep test" for the automatic testing of AVs using neural networks. In [35], an approach involving subjecting a controller to a dynamically chosen set of fault scenarios within a vehicle simulator to identify classes of vehicle faults and to produce noteworthy performance by the vehicle controller was proposed.

A recent body of research extensively covered various facets crucial to the reliability and safety of automated vehicles. One study [37] delved into verification and validation methods, highlighting their paramount importance in decision-making and planning within this domain. Another study [38] emphasized the significance of digital twins in enhancing the understanding and performance of connected and automated vehicles. Moreover, a separate research endeavor [39] focused on the pivotal role of virtual testing, demonstrating its efficacy through a longitudinal dynamics validation example. Additionally, insights from a study by [40] shed light on the application of dense reinforcement learning to bolster the safety validation of autonomous vehicles, showcasing advanced techniques for enhancing their safety. Lastly, [41] contributed valuable considerations regarding the determinism of game engines used in simulation-based verification for autonomous vehicles, ensuring accuracy and reliability in simulation environments.

III. Mathematical Background

In [21, 42], the authors presented a method to generate scenarios that can be grouped into equivalent classes. The objective was to address the completeness and coverage of a test scenario by identifying all its variations and equivalent scenarios. Hence, producing a collection of scenarios that can be combined to generate other scenarios. Using the equivalence class representatives, we can define *coverage*. To illustrate this, suppose an autonomous algorithm's response was tested using a representative of a class of scenarios. If the algorithm made the correct or expected decision then it would be considered as *passed* this class representative. Thus, it is expected to behave similarly, that is pass, in all the equivalent scenarios in that class. Thus, we consider that class of scenarios *covered*. The test results may be perceived as binary—pass or fail—without accounting for varying degrees of success or failure. Furthermore, the test output might be considered as probabilistic or a percentage, which influences the decision-making process for equivalent scenarios. With probabilistic outputs, decisions are not strictly binary

Downloaded from SAE International, Friday, January 31, 2025

but rather revolve around confidence scores or probability estimates.

In this context, the algorithm's decision on equivalent scenarios might be influenced by the confidence level it has in its prediction. For instance, if it's 80% confident that the response belongs to a certain category (such as range for safety distance or speed), it might make a decision based on that level of confidence, potentially treating scenarios with higher confidence differently than those with lower confidence scores.

So, while the autonomous algorithm's successful identification of a representative case within a class implies potential success for equivalent scenarios, the influence of probabilistic outputs means that decisions might vary based on the level of confidence in the algorithm's prediction. Hence having an accurate of the model is a prerequisite to obtaining equivalent classes and coverage.

For example, suppose an AV is to be tested on how it would approach and stop at a stop sign. In this case, one may consider all types of intersections that have stop signs as equivalent. Hence, if the AV slows down gradually and comes to a complete stop at the stop sign, then it will be able to do the same regardless of the type of intersection.

In this work, we will be using the same definitions as in [21, 42].

Definition 1. A scene vector $\mathbf{C}(k) \in \mathbb{R}^{n_i}$ is a vector whose components are the parameters that describe the environment surrounding the AV within $\mathcal{N}_k$ units of distance at discrete time intervals ($k = t_0 + n_k\,\Delta t$). The distance $\mathcal{N}_k$ will be called the *radius* of the scene vector and the parameters are grouped and organized into four main groups:

- The parameters describing the dynamics of the AV, or unit under test (UUT).
- The parameters describing the dynamics of moving actors.
- The parameters describing the constants or static components.
- Communication parameters between the UUT and any actor or the environment, such as visibility, weather conditions, road surface conditions, etc.

The next state is determined by a function ζ defined as:

Definition 2. Let ζ be a function for which the domain is the Cartesian product space of the space of scene vectors (i.e., a comprehensive space that holds all potential configurations or descriptions of scenes by combining different sets of scene vectors through a mathematical operation called the Cartesian product), communications from other actors—namely their velocities, positions, and directions—and the desired action of the ego, such as velocity and position. The range of ζ is the space of the scene vector. That is, if $\mathbf{C}(k)$ be the scene vector at time step k, then

$$\mathbf{C}(k+1) = \zeta\left(\mathbf{C}(k),\ \text{Ego's desired Input, actors' input}\right) \qquad \text{Eq. (1)}$$

where $\mathbf{C}(k+1)$ is the vector corresponding to the next time step calculated using the Newtonian laws of motion.

Thus, a *scene* or *scenario* χ was formally defined as a matrix whose columns are scene vectors at consecutive time steps:

$$\chi \triangleq \left[\mathbf{C}(0),\ \mathbf{C}(1),\ldots,\ \mathbf{C}(k)\right]. \qquad \text{Eq. (2)}$$

Definitions 1 and 2 enable one to define equivalence relations between scenes depending on the needs of the testing and verification process, furthering the flexibility of the proposed framework. Now, let's define the verification process.

Consider a scene as in Definitions 1 and 2. Notice that every actor—whether moving or stationary—and even the road structure enter the scene with a certain set of assertions. For example, another vehicle will assert that the distance between the UUT and it remains greater than the minimum safety distance. Based on this the authors in [21] defined a verification function based on the multiplication of matrices as follows:

Definition 3. Given a scenario χ, we define the **assertion function** $\mathcal{V}$ as a function with the domain being the set of scenario matrices and the range being the interval [0, 1], which has a predetermined set of weighted assertions. The output of this function is a probability (or a percentage) calculated as the weighted average based on the predetermined assertions where an output of 0 means the UUT fails and an output of 1 represents the UUT passing.

The verification function can be presented as follows:

Let χ be a scene given by a matrix of size $n \times (k+1)$ where k is the current time step. Now, assume that there are m assertions placed on χ. Let $\mathcal{A}$ be an $m \times n \times (k+1)$ multilayer matrix (a tensor) called the *assertion matrix*. Let $\mathcal{A}_j$ be the $m \times n$ matrix that corresponds to the jth layer of $\mathcal{A}$. In turn, $\mathcal{A}_j$ corresponds to the jth column (scene vector) in χ where each row represents an assertion and each column represents one of the parameters in that scene vector. In other words, the entries in each row are the weights representing the relation between the assertions and the parameters in the scene vectors. Next, the **assertion function** $\mathcal{V}$ is defined as follows:

$$\mathcal{V}(\chi) = \mathcal{A}\chi - \chi_{ref} \qquad \text{Eq. (3)}$$

Here, the product $\mathcal{A}\chi$ is an $(m \times 1 \times (k+1))$ tensor represented as a multilayered matrix in which the jth layer is $\mathbf{v}_j$ is the $(m \times (k+1))$ vector obtained by multiplying the jth layer of $\mathcal{A}$ by the jth column of χ. That is, $\mathbf{v}_j = \mathcal{A}_j\mathbf{C}(j)$. In addition, χ_{ref} is an $(m \times 1 \times (k+1))$ tensor represented as a multilayered matrix with each layer consisting of the vector of acceptable values of the parameters for each assertion for each time step, denoted by $\mathcal{C}_{0,j}, j = 0,\ldots,k$.

This grouping serves to consolidate all computational steps. Notably, the operations involving additions,

Downloaded from SAE International, Friday, January 31, 2025

subtractions, and products are well-defined matrix operations, ensuring the reproducibility of tests.

IV. Proposed Testing Framework

Here, we present and discuss the scenario generation and verification procedure for the proposed framework as illustrated in Figure 1.

Identify Initial Rules: This step establishes the fundamental characteristics of the scenario, defining key elements such as road infrastructure type and actor specifics (dynamic or static). For instance, detailing the nature of intersections or highways and determining actor types, behaviors, and counts.

Define Legal Road Structure: Expanding on initial rules, this stage involves specifying intricate details of the road system, including lane configurations, non-standard speed limits, traffic control mechanisms, and environment-specific features.

Neural Network-based Scene Generation: Leveraging sophisticated machine learning architectures, this phase utilizes extensive scene databases to generate initial position and velocity vectors for scenario actors. There are many machine learning structures that can be used such as recurrent (RNNs) and long–short-term memory (LSTM) neural networks, which are used to model sequences of data [43, 44]. In the context of scenario generation, these networks can analyze sequential data representing various driving scenarios to generate initial position and velocity vectors for actors (vehicles, pedestrians, etc.) in a scene. Additionally, using convolutional networks that are adept at extracting spatial features from images, or generative adversarial networks (GANs) that can be employed to generate synthetic scenes by learning the underlying distribution of real-world driving scenarios or a hybrid approach as done in [45, 46, 47]. Here, we propose the following neural network architecture setup.

Neural Network Structure and Training Process: Designing a neural network architecture involving:

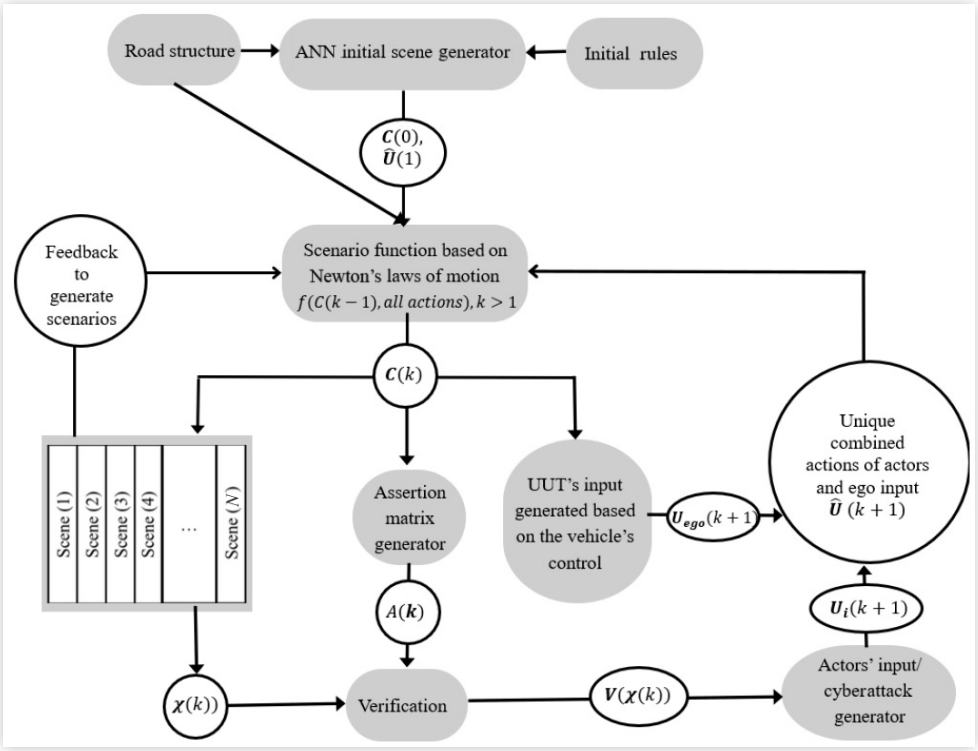
- Input layers representing scenario features.
- Multiple hidden layers employing activation functions (e.g., ReLU, sigmoid).
- An output layer generating initial condition vectors for actors and the UUT.

Training involves:

- Random weight initialization.
- Defining appropriate loss functions to measure error.
- Optimization techniques (e.g., gradient descent) to minimize loss.
- Backpropagation for parameter updates.

Validation, Feasibility Check, and Feedback Loop: Validation using separate datasets to ensure robustness and

FIGURE 1 Scenario generation and testing procedure.



© The Authors

Downloaded from SAE International, Friday, January 31, 2025

prevent overfitting. Feasibility checks to comply with physical laws, safety protocols, and environmental rules. An iterative feedback loop incorporates insights from infeasible scenarios for network refinement.

Scenario Generation Function Activation: Initiating subsequent scene creation based on initial vectors and established road structure, following Newtonian laws of motion. The function ζ, as defined in Definition 2, is based on the Newtonian laws of motion. The domain of ζ is the space of feasible scene vectors as well as $U(k)$, the combined actions of the actors, and the UUT input at every time step. Initially, $\mathbf{C}(0)$ along with the road structure are used as a seed to generate $\mathbf{C}(1)$, which is used to generate the rest of the scenes and to create the scenario χ.

For each time step $k > 1$, ζ uses the vector $\mathbf{C}(k)$ along with the road structure and $U(k)$ (the combined inputs of the actors and the ego (UUT)) to produce $\mathbf{C}(k)$. At this point, the vector $\mathbf{C}(k)$ is used for the following:

- As an input for the UUT control systems to generate the decision (desired input) of the UUT, $U_1(k)$, to be used for the $(k + 1)$th time step.
- Constructing the assertion matrix $A(k)$ to be used for assessing the decision made by the UUT at the kth time step.
- The scene is added as the last column in the scenario matrix χ.
- The scene is used as the input for ζ to produce the scene for the next time step.

Verification Function Execution: Scrutinizing the UUT assertions across all time steps using assertion matrices, identifying and quantifying UUT failures throughout the scenario. When the kth scene vector, $\mathbf{C}(k)$, is generated in the last step, a corresponding assertion matrix $A(k)$ will be generated. The verification function $\mathcal{V}$ uses, as its domain, the entire scenario χ and the assertion matrices $A(k)$ for each $k = 0, 1, \ldots, k$ to determine whether the UUT passed or failed any of the assertions for any of the time steps. The verification process is performed at each time step to identify where the UUT failed and the level of its failure.

Actor Input and Cyberattack Generation: Modifying actor actions ($U_i(k)$) based on verification results, potentially introducing simulated cyberattacks to test system resilience.

Class Equivalence and Coverage Check: Ensuring uniqueness of input vectors before computing subsequent scenes, preventing repetition or equivalence with previously tested scenarios. Here, the input from the actors and the UUT are combined into a single input vector $U(k)$ to be used in the computation of $\mathbf{C}(k + 1)$. However, before that is done, a check is performed to make sure that the resulting action has not been repeated nor has it been included in an equivalent scene previously tested. If the input is equivalent to a previously used vector then the actor input generator in the previous step must produce a new vector.

A. The Structure of the Assertion Matrix A

For a scene with l actors, the scene vector may be defined as a $(6 + 8l) \times 1$ column matrix. The first six rows list the dynamics of the UUT with each other actor having eight additional rows to capture the dynamics as well as the longitudinal and latitudinal distance from the UUT.

Here, the corresponding assertion matrix $\mathcal{A}$ will be defined as an $m \times n$, where $m = 1 + 2l$ and $n = 6 + 8l$. The first row of $\mathcal{A}$ is defined, as in 6, such that it can be used to calculate the speed (or the square of the speed) of the UUT and compare it with the speed limit. Furthermore, for each moving actor, the UUT will pass if it maintains the minimum safety distance.

Therefore, the matrix $\mathcal{A}$ will have the following form:

$$\begin{bmatrix} 0 & 0 & \dot{x}(t) & \dot{y}(k) & 0 & \ldots\ldots & & \\ & & & & -1 & 0 & & \\ & & \mathbf{0} & & 0 & -1 & \mathbf{0} & \cdots \\ & & & & & & -1 & 0 \\ & & \mathbf{0} & & \mathbf{0} & & 0 & -1 \\ & & \vdots & & \vdots & & \vdots & \ddots \end{bmatrix}_{m \times n}$$

The first negative identity matrix corresponding to the first actor begins at the $(2, 6 + 6 + 1) = (2, 13)$ entry, the second will appear at the $(4, 13 + 8) = (4, 21)$, the third will be at $(6, 21 + 8) = (6, 29)$. That is, the negative identity submatrix that will correspond to actor i will be at the entry $(2i, 13 + 8(i - 1))$.

Remark 1. An important point to make that all the vehicle's decisions will effect its dynamics. For instance, if the UUT was approaching a red traffic light, then the passing (correct) response would be to slow down and come to a complete stop at an appropriate position. Then notice that the *correctness* of the decision can be measured by measuring the speed of the UUT and relative distance between the UUT and actors in the scene, including the traffic light.

Remark 2. Furthermore, note that we assumed that the UUT has an accurate model of the environment around it. That is, the previous example we assume that the UUT could recognize a red traffic light and we were testing its response.

B. Illustration

Suppose we had a scene where there was a stretch of road with a speed limit of 30 mph and only the UUT and another actor. Then, for this illustration, we may ignore the latitudinal coordinates and only consider the longitudinal ones. Thus, we could set up a scene vector at a time step k as

$$\mathbf{C}(t) = \left[x(t),\ \dot{x}(t),\ \ddot{x}(t),\ x_1(t),\ \dot{x}_1(t),\ \ddot{x}_1(t),\ d_1(t)\right]^T \quad \text{Eq. (4)}$$

Downloaded from SAE International, Friday, January 31, 2025

$$\mathbf{C}_{\bar{\beta}}(t)=\left[\beta_{1,1}(t),\ \beta_{1,2}(t),\ldots\right]^{T} \quad \text{Eq. (5)}$$

where $x(t)$, $v(t)=\dot{x}(t)$, and $a(t)=\dot{v}(t)=\ddot{x}(t)$ are defined to be the position, velocity, and acceleration of the UUT and similarly x_1, v_1, and $a_{1\ are}$ defined to be the position, velocity, and acceleration of the actor. Finally, $d_1(t)$ is the distance between the actor and the UUT. $C_{\bar{\beta}}$ is the injected attack between the UUT and other actors, which will be referenced in Section V. For instance, $C_{\bar{\beta}}$ will be injected fault, injected delay for FDI attacks and time-delay switch (TDS) attacks, respectively. Furthermore, we can define $C_{\bar{\beta}}$ to represent other types of attacks such as denial of service (DoS) attack.

The corresponding assertion matrix $\mathcal{A}$ will be a 3 × 14:

$$\mathcal{A}(t)=\begin{bmatrix}0 & 1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & -1\end{bmatrix} \quad \text{Eq. (6)}$$

Here, we will have:

$$\mathcal{A}\mathbf{C}(t)=\begin{bmatrix}\left(\dot{x}(t)\right)\\ -d_1(t)\end{bmatrix} \quad \text{Eq. (7)}$$

Next, we can apply the verification function as follows:

$$\begin{aligned}\mathcal{V}\left(\mathbf{C}(t)\right)&=\mathcal{A}\mathbf{C}(t)-\mathcal{C}_{ref}\\ &=\begin{bmatrix}\dot{x}\\ -d_1(t)\end{bmatrix}-\begin{bmatrix}Speed_{ref}\\ -d_{min,x}(t)\end{bmatrix}\end{aligned} \quad \text{Eq. (8)}$$

where $d_{min,x}(t)$ is the minimum safety longitudinal distances depending on the speed of both the UUT and the actor and their positions. The UUT would "*pass*" the test if the result of Equation 8 is a vector with non-positive entries in all its rows.

The vector $\mathcal{C}_{ref}$ is *the reference vector*. $\mathcal{C}_{ref}$ is to be generated automatically and contain the speed limit from the road structure input and the minimum safety distances computed via rules listed in [48].

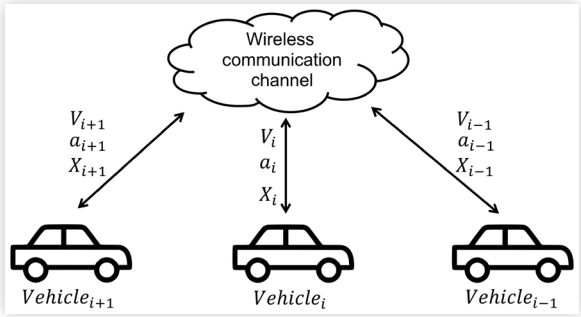
V. CACC Algorithm

CACC is just one example of the many ADAS that utilizes connectivity between nearby vehicles to maintain speed while ensuring safe following distances [49]. CACC was chosen for testing in our framework as the majority of existing algorithms have been designed assuming that their communication channels are secure. However, the mere implementation of wireless communication renders CACC prone to various attacks, such as FDIs.

A. Mathematical Model of CACC under FDI Attacks

The wireless connectivity is integrated as part of the system model, implemented as a feed-forward signal that relays information through wireless communication. Figure 2 illustrates the data flow between CACC vehicles. For a string of homogeneous vehicles with CACC capabilities following a leader using a dynamic velocity profile, the dynamics model of vehicles are described as

FIGURE 2 Vehicle CACC diagram.



$$\begin{cases}\dot{x}_i(t)=v_i(t)\\ \dot{v}_i(t)=-\dfrac{b_i}{m_i}v_i(t)+u_i(t)+d_i(t)\end{cases} \quad \text{Eq. (9)}$$

where $i \in \{1,\ldots,n\}$ denotes the follower vehicles, n is the maximum number of follower vehicles, and $i-1$ indicates the leader vehicle. In Equation 9, $m_i \in \mathbb{R}$ is the vehicle's mass and $b_i \in \mathbb{R}$ is the friction force between the road and tires. Also $x_i \in \mathbb{R}$, $v_i \in \mathbb{R}$, $u_i \in \mathbb{R}$, and $d_i \in \mathbb{R}$ represent the position, velocity, control input, and external disturbance, respectively. Furthermore, the parameters $\dot{x}_i$ and $\dot{v}_i = \ddot{x}_i$, for all i, form (9) will be the entries of the scene vector in (4).

Assumption 1. The disturbance is assumed to be bounded and continuous by a known constant, such that $\|d_i(t)\| < \bar{d}_i$ for $t \geq t_0$, where $\bar{d}_i \in \mathbb{R}_+$ [50].

B. FDI Attack Representation

The FDI attacks are injected into the wireless communication network of the connected vehicles, resulting in the vehicles acting upon corrupted data. This causes instability in the vehicle platoon, increasing the likelihood of collisions. In this model, we assume that acceleration is the parameter affected by the attack, interpreted as Equation 10. The attack affects the output, which transforms it into the observed output

$$\pi_i\left(a_{i-1}(t)\right)\triangleq a_{i-1}(t)+\beta_i(t) \quad \text{Eq. (10)}$$

where $\pi_i \in \mathbb{R}$ is the attack function, $\beta_i \in \mathbb{R}$ is the bounded, unknown, continuous, and time-varying FDI attack, and a_{i-1} is the leader's acceleration.

Assumption 2. The FDI attack is assumed to be bounded and differentiable such that $|\beta_i(t)| \leq \bar{\beta}_i$, where $t \geq t_0$ and $\bar{\beta}_i$ is a positive constant [51, 52].

Downloaded from SAE International, Friday, January 31, 2025

C. Resilient Controller Design

To measure the accuracy of the FDI attack estimation, the estimation error for the FDI attack, $\tilde{\beta}_i : [t_0, \infty) \rightarrow \mathbb{R}^{n_i}$, is defined as

$$\tilde{\beta}_i(t) \triangleq \beta_i(t) - \hat{\beta}_i(t) \quad \text{Eq. (11)}$$

where $\hat{\beta} \in \mathbb{R}$ is the estimation of FDI attack.

To quantify the accuracy of the controller design, we defined the tracking error signal as

$$e_i(t) \triangleq x_i(t) - x_{i-1}(t) + D_i + x_{d_i}(t) \quad \text{Eq. (12)}$$

where $x_{d_i} \in \mathbb{R}$ denotes the desired distance between vehicles and $D_i \in \mathbb{R}$ is the length of ith vehicle.

To facilitate the design process, the second error signal, r_i, is defined as

$$r_i(t) \triangleq \dot{e}_i(t) + \alpha_i e_i(t) \quad \text{Eq. (13)}$$

where $\alpha_i \in \mathbb{R}_{>0}$ is an user-specified gain.

To facilitate the stability analysis, we need to define another auxiliary error signal $\tilde{r}_{i-1} \in \mathbb{R}$ as

$$\tilde{r}_{i-1}(t) \triangleq \dot{\tilde{x}}_{i-1}(t) + \alpha_{i-1}\tilde{x}_{i-1}(t) \quad \text{Eq. (14)}$$

where $\alpha_{i-1} \in \mathbb{R}_{>0}$ is a gain defined by users.

1. *Controller Design*: The control signal was designed using a Lyapunov stability analysis to ensure that the system remains stable under FDI attacks. The control signal is defined as

$$u_i(t) \triangleq \frac{b_i}{m_i}v_i(t) - \frac{b_{i-1}}{m_{i-1}}v_{i-1}(t) + \bar{u}_{i-1} - \hat{\beta}_i(t) - \ddot{x}_{d_i}(t) - \alpha_i r_i(t) + \alpha_i^2 e_i(t) - e_i(t) - K_{1_i} r_i(t) \quad \text{Eq. (15)}$$

where $K_{1_i} \in \mathbb{R}_{>0}$ is an user-specified gain, $\bar{u}_{i-1} \triangleq u_{i-1} + \beta_i$, and $u_{i-1} \in \mathbb{R}$ is the actual control signal of the leader.

2. *FDI Attack Estimation*: Considering respect to the spatial domain, the NN estimation of FDI attack can be described as

$$\hat{\beta}_i \triangleq \hat{W}_i^T \sigma\left(\hat{V}_i^T \delta_i\right) \quad \text{Eq. (16)}$$

where $\hat{W}_i \in \mathbb{R}^{(n_i+1)\times n_i}$, $\hat{V}_i \in \mathbb{R}^{(n_i+1)\times n_n}$ represent the estimated ideal weights, n_n is the number of neurons in the hidden layer, $\sigma(\cdot) \in \mathbb{R}^{(n_n+1)}$ is an activation functions vector, and δ_i is given as

$$\delta_i \triangleq \left[1, \ \hat{\beta}_i^T\right]^T \quad \text{Eq. (17)}$$

Resulting from the stability analysis, the updating laws for the NN weights are described as

$$\dot{\hat{W}}_i = proj\left(\Gamma_{1_i}\left(\hat{V}_i^T \delta_i\right)\phi_i\right) \quad \text{Eq. (18)}$$

and

$$\dot{\hat{V}}_i = proj\left(\Gamma_{2_i}^T \phi_i \hat{W}_i^T \sigma\left(\hat{V}_i^T \delta_i\right)\right) \quad \text{Eq. (19)}$$

where $\phi_i \triangleq r_i - \tilde{r}_{i-1}$ while Γ_{1_i}, $\Gamma_{2_i} \in \mathbb{R}^{n_i \times n_i}$ are definite positive matrices, and the function $proj(\cdot)$ denotes the Lipschitz continuous projection operator defined in [50].

3. *Observer Design*: The observer is designed in such a way to ensure that the system remains stable. Based on the stability analysis in next subsection, the observer for vehicle $i-1$ is designed as

$$\ddot{\hat{x}}_{i-1}(t) = -\frac{b_{i-1}}{m_{i-1}}v_{i-1}(t) + \bar{u}_{i-1}(t) - \hat{\beta}_i + L_{1_i}\tilde{r}_{i-1} + \alpha_{i-1}\tilde{r}_{i-1} - \alpha_{i-1}^2\tilde{x}_{i-1} + \tilde{x}_{i-1} \quad \text{Eq. (20)}$$

where $L_{1i} \in \mathbb{R}_{>0}$ and $\alpha_{i-1} \in \mathbb{R}_{>0}$ are user-defined gains and $\tilde{x}_{i-1} : [t_0, \ \infty) \rightarrow \mathbb{R}$ is the state estimate error, which is defined as

$$\tilde{x}_{i-1}(t) \triangleq x_{i-1}(t) - \hat{x}_{i-1}(t) \quad \text{Eq. (21)}$$

where $\hat{x}_{i-1} \in \mathbb{R}$ is the estimated position for the lead vehicle.

D. Stability Analysis

Let $V_i : \mathrm{R}^5 \times [0, \infty) \rightarrow \mathrm{R}_{\geq 0}$, a radially unbounded, positive definite, continuously differentiable Lyapunov function defined as

$$V_{L_i} = \frac{1}{2}\tilde{x}_{i-1}^2 + \frac{1}{2}\tilde{r}_{i-1}^2 + \frac{1}{2}e_i^2 + \frac{1}{2}r_i^2 + H_{L_i} \quad \text{Eq. (22)}$$

where $H_{Li} \in \mathrm{R}_{\geq 0}$ is written as

$$H_{L_i} \triangleq \frac{1}{2}tr\left(\tilde{W}_i^T \Gamma_{1_i}^{-1}\tilde{W}_i\right) + \frac{1}{2}tr\left(\tilde{V}_i^T \Gamma_{2_i}^{-1}\tilde{V}_i\right) \quad \text{Eq. (23)}$$

where $\tilde{V}_i = V_i - \hat{V}_i$ is the inner NN weight error and $\tilde{W}_i = W_i - \hat{W}_i$ is the outer NN weight error. Since we used projection operators to design $\hat{W}_i$ and $\hat{V}_i$, therefore, $\tilde{W}_i$ and $\tilde{V}_i$ are bounded, and subsequently H_{L_i} is bounded by $|H_{L_i}| \leq H_{L_i,max}$ where $H_{L_i,max} \in \mathbb{R}_{>0}$. Let $\pi_i \in \mathbb{R}^{2ni}$ be define as

$$q_i \triangleq \left[\tilde{r}_{i-1}^T, \ \tilde{x}_{i-1}^T, e_i^T, \ r_i^T\right]^T \quad \text{Eq. (24)}$$

and let ψ_{1_i} and ψ_{2_i} be defined as

$$\psi_{1_i} \triangleq \frac{1}{2}\|q_i\|^2 \quad \text{Eq. (25)}$$

Downloaded from SAE International, Friday, January 31, 2025

and

$$\psi_{2_i} \triangleq \|q_i\|^2 \quad \text{Eq. (26)}$$

therefore, the Lyapunov function satisfies the following inequality

$$\psi_{1_i} \le V_{L_i} \le \psi_{2_i} + H_{L_i,\max} \quad \text{Eq. (27)}$$

Taking the derivative of (22) yields

$$\begin{aligned}\dot{V}_{L_i} &= \tilde{x}_{i-1}\dot{\tilde{x}}_{i-1} + \tilde{r}_{i-1}\dot{\tilde{r}}_{i-1} + e_i\dot{e}_i + r_i\dot{r}_i \\ &-tr\left(\tilde{W}_i\Gamma_{1i}^{-1}\dot{\hat{W}}_i\right) - tr\left(\tilde{V}_i\Gamma_{2i}^{-1}\dot{\hat{V}}_i\right)\end{aligned} \quad \text{Eq. (28)}$$

Substituting (13), (14), and their time derivatives into (28) results in the following

$$\begin{aligned}\dot{V}_{L_i} &= \tilde{x}_{i-1}\left(\tilde{r}_{i-1} - \alpha_{i-1}\tilde{x}_{i-1}\right) \\ &+ \tilde{r}_{i-1}\left(-L_{1_i}\tilde{r}_{i-1} - \tilde{x}_{i-1} - \tilde{\beta}_i\right) \\ &+ e_i\left(r_i - \alpha_i e_i\right) + r_i\left(\tilde{\beta}_i - K_{1_i}r_i - e_i + d_i\right) \\ &- tr\left(\tilde{W}_i\Gamma_{1i}^{-1}\dot{\hat{W}}_i\right) - tr\left(\tilde{V}_i\Gamma_{2i}^{-1}\dot{\hat{V}}_i\right)\end{aligned} \quad \text{Eq. (29)}$$

Knowing that a Taylor's series approximation can be applied to (21), we have

$$\tilde{\beta}_i = \tilde{W}_i^T\sigma\left(\hat{V}_i^T\delta_i\right) + \hat{W}_i^T\sigma'\left(\hat{V}_i^T\delta_i\right)\tilde{V}_i^T\delta_i + N_{n_i} \quad \text{Eq. (30)}$$

where

$$N_{n_i} \triangleq \tilde{W}_i^T\sigma'\left(\hat{V}_i^T\delta_i\right)\tilde{V}_i^T\delta_i + W_i^T\vartheta\left(\tilde{V}_i^T\delta_i\right) + \gamma_i \quad \text{Eq. (31)}$$

where ϑ denotes higher order terms and N_{n_i} is bounded such that $\|N_{n_i}\| \le \bar{n}_{n_i}$, where $\bar{n}_{n_i} \in \mathbb{R}_{>0}$.

Substituting (30) in (29) results

$$\begin{aligned}\dot{V}_{L_i} &= -\alpha_i e_i^2 - \alpha_i\tilde{x}_{i-1}^2 - K_{1_i}r_i^2 - L_{1_i}\tilde{r}_{i-1}^2 + r_i d_i \\ &+ \phi_i\left(\tilde{W}_i^T\sigma\left(\hat{V}_i^T\delta_i\right) + \hat{W}_i^T\sigma'\left(\hat{V}_i^T\delta_i\right)\tilde{V}_i^T\delta_i + N_{n_i}\right) \\ &- tr\left(\tilde{W}_i\tilde{\mathrm{A}}_{1i}^{-1}\dot{\hat{W}}_i\right) - tr\left(\tilde{V}_i\tilde{\mathrm{A}}_{2i}^{-1}\dot{\hat{V}}_i\right)\end{aligned} \quad \text{Eq. (32)}$$

Substituting and updating laws (18) and (19) in (32) cancels the NN terms. By applying Young's inequality to select terms in (32) results

$$\begin{aligned}r_i N_{n_i} &\le \frac{1}{2\varepsilon_0}\|r_i\|^2 + \frac{\varepsilon_0}{2}\|N_{n_i}\|^2 \\ \tilde{r}_{i-1}N_{n_i} &\le \frac{1}{2\varepsilon_1}\|\tilde{r}_{i-1}\|^2 + \frac{\varepsilon_1}{2}\|N_{n_i}\|^2 \\ r_i d_i &\le \frac{1}{2\varepsilon_2}\|r\|_i^2 + \frac{\varepsilon_2}{2}\|d_i\|^2\end{aligned} \quad \text{Eq. (33)}$$

where ε_0, ε_1, and ε_2 are positive known constants.

Applying Young's inequality, the Equation 32 becomes

$$\begin{aligned}\dot{V}_{L_i} &\le -\alpha_{i-1}\|\tilde{x}_{i-1}\|^2 - L_{1_i}\|\tilde{r}_{i-1}\|^2 - \alpha_i\|e_i\|^2 - K_{1_i}\|r_i\|^2 \\ &+ \frac{1}{2\varepsilon_0}\|r_i\|^2 + \frac{1}{2\varepsilon_1}\|\tilde{r}_{i-1}\|^2 + \frac{1}{2\varepsilon_2}\|r_i\|^2 + \varphi_i\end{aligned} \quad \text{Eq. (34)}$$

where φ_i is defined as

$$\varphi_i \triangleq \frac{\varepsilon_0}{2}\bar{n}_{n_i}^2 + \frac{\varepsilon_1}{2}\bar{n}_{n_i}^2 + \frac{\varepsilon_2}{2}\bar{d}_i^2 \quad \text{Eq. (35)}$$

Re-arranging (34) results in

$$\begin{aligned}\dot{V}_{L_i} &\le -\left(\alpha_{i-1}\right)\|\tilde{x}_{i-1}\|^2 - \left(\alpha_i\right)\|e_i\|^2 \\ &-\left(L_{1_i} - \frac{1}{2\varepsilon_1}\right)\|\tilde{r}_{i-1}\|^2 \\ &-\left(K_{1_i} - \frac{1}{2\varepsilon_0} - \frac{1}{2\varepsilon_2}\right)\|r_i\|^2\end{aligned} \quad \text{Eq. (36)}$$

Let the sufficient conditions be defined as

$$\begin{aligned}\alpha_{i-1} &> 0 \\ \alpha_i &> 0 \\ L_{1_i} &> \frac{1}{2\varepsilon_1} \\ K_{1_i} &> \left(\frac{1}{2\varepsilon_0} + \frac{1}{2\varepsilon_2}\right)\end{aligned} \quad \text{Eq. (37)}$$

Given the sufficient conditions in (37), α_{1_i} and α_{2_i} can be defined as

$$\alpha_{1_i} \triangleq L_{1_i} - \frac{1}{2\varepsilon_1} \quad \text{Eq. (38)}$$

$$\alpha_{2_i} \triangleq K_{1_i} - \frac{1}{2\varepsilon_0} - \frac{1}{2\varepsilon_2} \quad \text{Eq. (39)}$$

From inequality (27), we know that the Lyapunov function is bounded, therefore, (36) can be written as

$$\dot{V}_{L_i} \le -\frac{\alpha_{3_i}}{\psi_{2_i}}V_{L_i} + \frac{\alpha_{3_i}}{\psi_{2_i}}H_{L_i,max} + \varphi_i \quad \text{Eq. (40)}$$

where $\alpha_{3_i} \triangleq \min\{\alpha_{i-1}, \alpha_i, \alpha_1, \alpha_2\}$. Given the sufficient equations provided in (37) are satisfied, the result in (40) ensures semi-globally uniformly bounded tracking.

VI. Implementation and Results

In this section, we intend to delve deeper into the validation and fine-tuning of these parameters. We will demonstrate

Downloaded from SAE International, Friday, January 31, 2025

how variations in these design parameters impact the framework's ability to adapt, detect faults, and maintain safe system behavior. This validation process will provide insights into the optimal parameter configurations that maximize the framework's performance across diverse testing scenarios. To that end, the procedure will be initiated manually without the use of a neural network.

To test the proposed verification framework, a prototype was developed using MATLAB/Simulink. The framework was implemented as a Simulink model that leverages several MATLAB functions to facilitate, both the verification framework and the simulation environment. The verification framework processes data generated by the simulation at each time step in order to score the performance of the UUT. The simulation environment is capable of modeling both environmental factors and actor trajectories. At this stage, the framework is capable of conducting test scenarios, recording the results and relevant data generated during the simulation, and enabling real-time visualization of testing.

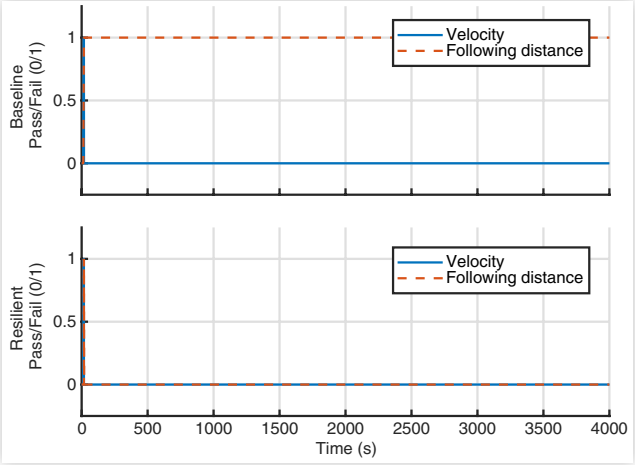
The simulation environment used in this implementation models are two actors, a leader and the follower. The lead vehicle operates independently of the follower vehicle, taking desired speed inputs from the test scenario. The follower vehicle acts as the UUT and possesses a controller that uses the leader's acceleration and position to maintain a desired following distance.

To leverage our simulation environment, a scenario generation script was developed. The script is customizable, taking several parameters as arguments. These are the maximum FDI attack values β_i, the maximum speed of the actor, the time constant as defined in the actor model, the total simulation time, and the desired sampling time. These parameters are then used to create a vector of random desired velocities at evenly spaced intervals for the desired simulation duration. The period between speed changes is derived as $3\tau_{i-1}$, where τ_{i-1} is the leader time constant, as utilized in the actor mathematical model, and ensures that the feasible scenarios are generated.

For testing the viability of the framework, test scenarios were developed. Each scenario was generated with a sample time of 0.01 and lasts for 4,000 seconds of simulated time. In all of the following scenarios, a random desired speed, between 0 and the speed limit, is sent to the lead vehicle's trajectory every 16.5 seconds.[1] This test vector is recorded as a three-dimensional matrix containing desired velocities, the injected FDI attack magnitude, and corresponding times in simulation. Using this standardized test case, the framework enables point-to-point comparisons of each CACC model throughout each simulation step. In the following scenarios, the verification process will classify the outcome as a failure if the distance between vehicles falls below the established minimum safe distance threshold. Furthermore, a failure in terms of velocity will be recorded when velocity of the following vehicle is more than the speed limit.

[1] The time constant of the lead vehicle is 5.5 seconds. Therefore, it's reasonable to generate random desired speed every 16.5 seconds to allow enough time to the velocity of lead vehicle to reach to 95% of its final value.

FIGURE 3 Scenario 1 verification check: baseline controller (top) and resilient controller (bottom).



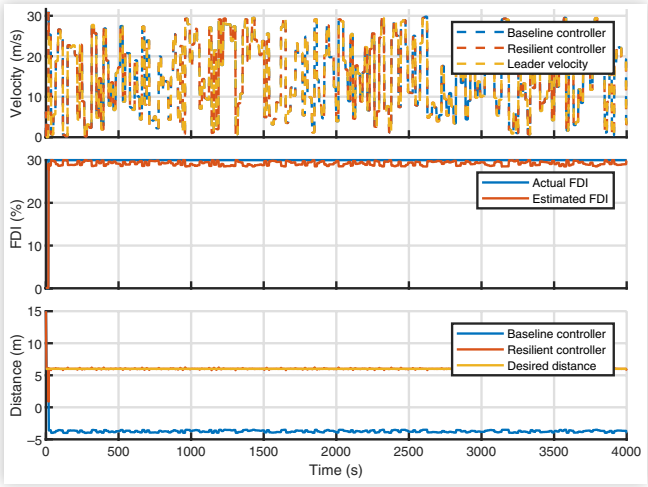
© The Authors

A. Scenario 1

1. *Description*: The first test scenario consisted of cycles of random desired speeds with a single FDI attack of 30% injected and maintained throughout testing.
2. *Results*: As shown in Figure 3, the controller equipped with detection was able to maintain a safe follow distance throughout testing. The baseline controller, on the other hand, spent the vast majority of the test colliding with the lead vehicle due to the attack.

Figure 4 demonstrates the performance of the resilient controller by presenting each actor's velocity, the FDI attack estimate and true value, as well as the distance between vehicles. As shown, an attack of 30% is injected and held constant throughout testing. The neural network detects the

FIGURE 4 Scenario 1 performance: resilient controller's velocity profile (top), FDI estimation performance (middle), and following distance of both controllers (bottom).



© The Authors

Downloaded from SAE International, Friday, January 31, 2025

attack and trains upon the signal to ensure adequate tracking to prevent an unsafe following distance. The baseline controller, however, is unable to detect and counter this attack, causing the follower to pass the leader vehicle.

Utilizing the framework and testing results, the controller parameters were tuned to improve performance. One area for improvement was in ensuring the follower vehicle followed the speed limit.

B. Scenario 2

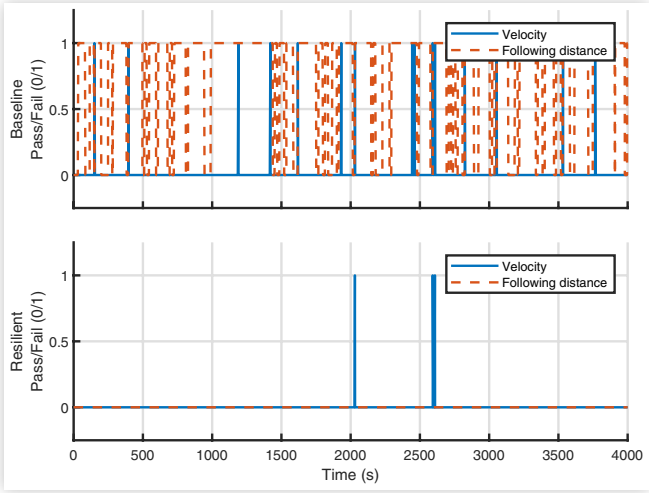
1. *Description*: In the second scenario, random speeds were periodically generated with random FDI attacks injected at each speed change.
2. *Results*: As shown in Figure 5, the resilient controller performed well on this test case, maintaining a safe following distance throughout the majority of testing and preventing any collision. The baseline controller, on the other hand, frequently followed the lead vehicle too closely, which resulted in several crashes. The data in Figure 6 displays the ability of the FDI estimator to adapt to the varying signal.

From the performance on the second scenario, the controller was able to be further tuned. In the second scenario, the neural network's parameters were adjusted to improve FDI estimation to maintain a safe following distance throughout the entirety of testing.

C. Scenario 3

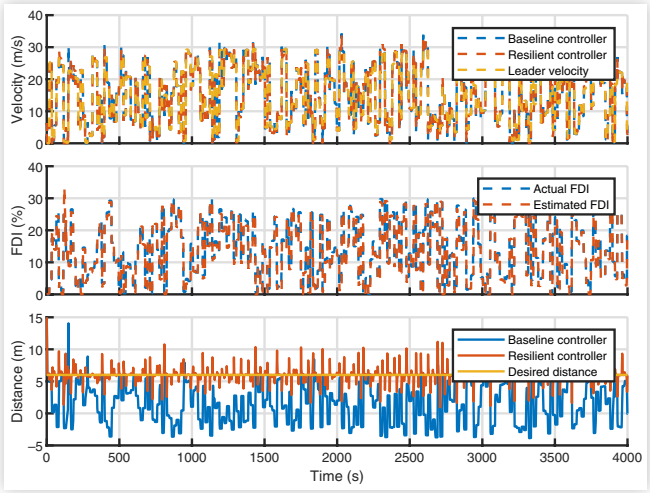
1. *Description*: In the third and final test, cycles of accelerations and decelerations to random speeds were generated with random FDI attacks injected at each deceleration. In addition, the acceleration of the leader is transmitted with additional white noise.

FIGURE 5 Scenario 2 verification check: baseline controller (top) and resilient controller (bottom).



© The Authors

FIGURE 6 Scenario 2 performance: resilient controller's velocity profile (top), FDI estimation performance (middle), and following distance of both controllers (bottom).

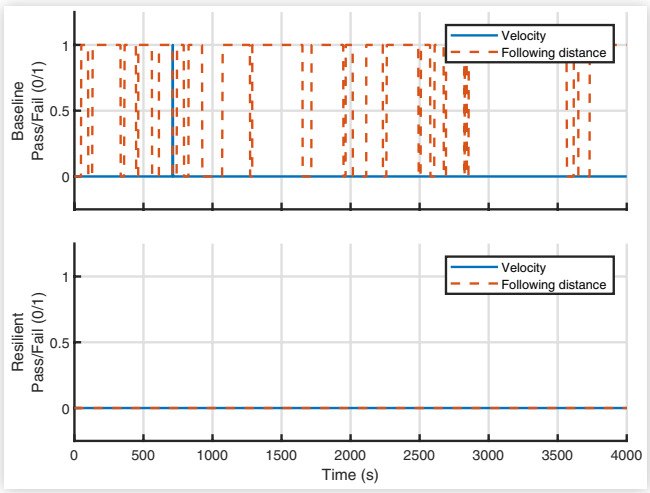


© The Authors

2. *Results*: The results are contained within Figure 7. As can be seen, the baseline controller encountered specific speed changes and attack injection combinations that resulted in an unsafe following distance at multiple points throughout the test. While the baseline controller performed drastically better than the baseline counterpart. The fact that an unsafe scenario was discovered for this baseline controller demonstrates the validity and importance of a framework that adaptively tests a system based on its performance.

The results of the final test show that our controller demonstrates a higher level of safety than the baseline controller. However, it is also evident that the controller could be refined for even further performance improvements.

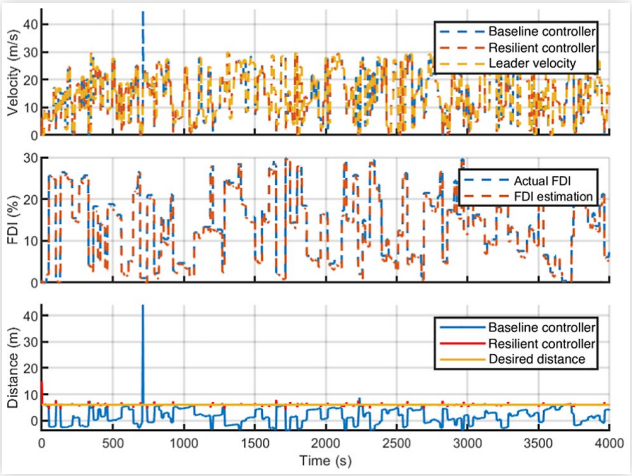
FIGURE 7 Scenario 3 verification check: baseline controller (top) and resilient controller (bottom).



© The Authors

Downloaded from SAE International, Friday, January 31, 2025

FIGURE 8 Scenario 3 performance: resilient controller's velocity profile (top), FDI estimation performance (middle), and following distance of both controllers (bottom).



© The Authors

This process of further testing and refinement is planned and discussed in our future works (Figure 8).

Remark 3. **Parameter Design and Framework Convergence.**

The design of parameters such as control gains, observe gains, and neural network structures significantly influences the convergence and overall performance of the testing framework. These parameters play a crucial role in determining the adaptability and robustness of the controller and estimator modules within the framework.

The control gains, for instance, regulate the response characteristics of the controller, affecting how quickly and accurately the system adjusts to varying scenarios. Similarly, the observe gains impact the convergence rate of the estimator module, influencing its ability to detect and adapt to injected faults or disturbances.

Moreover, the neural network structures, including architecture, layer configurations, and learning algorithms, dictate the learning capacity of the system. Properly designed neural network structures ensure that the system can effectively recognize and respond to abnormal scenarios, contributing to the overall convergence of the testing framework. The gains of controller, observer, and FDI estimator can be initialized based on the sufficient conditions given in (37) and optimization algorithms such as genetic algorithms can be used to find the best parameters.

D. Threat Model and Risk Analysis

This section develops and assesses a threat model following the ISO/SAE 21434: Road Vehicles—Cybersecurity Engineering standard, focusing on a resilient CACC algorithm used in two vehicles on a straight highway. Risks linked to the identified threats can be evaluated to compare the baseline and resilient controller. This article uses the risk formula from [53]:

$$R = 1 + F \times I \qquad \text{Eq. (41)}$$

where R represents the risk value, F quantifies aggregated attack feasibility rating (very low = 0, low = 1, medium = 1.5, high = 2), and I denotes impact rating (negligible = 0, moderate = 1, major = 1.5, severe = 2). For this work, impact is measured as

$$I = 2 \times \frac{C}{A} \qquad \text{Eq. (42)}$$

where C and A denote the number of crashes and attacks, respectively. The ratio is scaled by two to match the impact range. Since a CACC system is critical, the impact I could be severe if the feature is disrupted. But for this research, a dynamic I is more useful for tuning the proposed controller. Attack feasibility is assumed to be high, because the test scenarios have a faulty leader vehicle signal.

The results of all three test scenarios are compiled in Table 1. The risk as well as RMSE of following distance and FDI estimation are presented for each controller. In all three tests, the baseline controller exhibited drastically higher risk than the resilient controller. In addition, the following distance error for the resilient controller is much less than the baseline controller.

VII. Conclusions and Future Work

The use of a computation model for testing and verifying the security of CAVs is a valuable tool in ensuring the safety and

TABLE 1 Testing results of baseline and resilient controller expressed in terms of risk and RMSE of following distance and FDI estimation.

	Baseline controller			Resilient controller		
Test scenario	Risk	Distance	FDI est.	Risk	Distance	FDI est.
1	796,459	9.7248	-	1	0.089	0.9746
2	274,175	5.4087	-	1	0.1875	2.6142
3	234,099	5.4517	-	1	0.1451	1.8331

© The Authors

Downloaded from SAE International, Friday, January 31, 2025

reliability of these systems. This article proposed a novel framework for testing and verifying the security of CAVs under attacks. In addition, we discussed a new and original secure CACC algorithm that combines model and learning-based techniques to detect and mitigate FDI attacks in real-time. The proposed framework was implemented in a software-in-the-loop environment to test the security of CACC under attacks. We showed that the proposed framework could demonstrate unsafe situations. As CAVs become increasingly prevalent, it is crucial that their security is thoroughly tested and verified to ensure the safety of passengers and other road users.

Despite all of the advantages of the resilient controller, it requires an accurate model of the leader and follower for the observer design. In addition, tuning the parameters of the developed controller is timely. Moreover, the resilient controller cannot compensate for attacks such as TDS attack. To improve the performance of the proposed framework, we plan to develop an adaptive algorithm to adjust the parameters of controllers in a closed-loop manner to refine their performance during testing. Furthermore, we will use machine learning algorithms to generate edge cases in our future work.

Acknowledgement

Partial support of this research was provided by the National Science Foundation under Grant No. ECCS-EPCN-2241718 and CNS-1919855. Any opinions, findings, and conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the sponsoring agency.

Contact Information

Ala J. Alnaser, PhD
ala.aj.alnaser@gmail.com
aalnaser@floridapoly.edu

References

1. Wang J.-S., "Target Crash Population for Crash Avoidance Technologies in Passenger Vehicles," Tech. Rep., 2019.
2. Singh S., "Critical Reasons for Crashes Investigated in the National Motor Vehicle Crash Causation Survey," DOT HS 812 115, Tech. Rep., NHTSA's National Center for Statistics and Analysis, Washington, DC, 2015.
3. Sargolzaei, A., "Security of Cyber-Physical Systems," MDPI-Multidisciplinary Digital Publishing Institute, 2022.
4. Liu, L., Liu, S., and Shi, W., "4C: A Computation, Communication, and Control Co-Design Framework for CAVs," *IEEE Wireless Communications* 28, no. 4 (2021): 42-48, doi:10.1109/MWC.201.2000512.
5. Rathore, R.S., Hewage, C., Kaiwartya, O., and Lloret, J., "In-Vehicle Communication Cyber Security: Challenges and Solutions," *Sensors* 22, no. 17 (2022): 6679.
6. Khalil, K., Eldash, O., Kumar, A., and Bayoumi, M., "Machine Learning-Based Approach for Hardware Faults Prediction," *IEEE Transactions on Circuits and Systems I: Regular Papers* 67, no. 11 (2020): 3880-3892, doi:10.1109/TCSI.2020.3010743.
7. Hirsch, T., "A Fault Localization and Debugging Support Framework Driven by Bug Tracking Data," arXiv preprint arXiv:2103.02386, 2021, https://www.frontiersin.org/articles/10.3389/fpubh.2020.00014/full.
8. Hayes, K., Blashki, G., Wiseman, J., Burke, S. et al., "Climate Change and Mental Health: Risks, Impacts and Priority Actions," *International Journal of Mental Health Systems* 12 (2018): 28, doi:https://doi.org/10.1186/s13033-018-0210-6.
9. Reason, J., "Human Error: Models and Management," *BMJ* 320 (2000): 768-770, doi:10.1136/bmj.320.7237.768.
10. Ahmed, A.-S.K. and Pathan, M., "False Data Injection Attack (FDIA): An Overview and New Metrics for Fair Evaluation of Its Countermeasure," *Complex Adaptive Systems Modeling* 8 (2020): 4, doi:https://doi.org/10.1186/s40294-020-00070-w.
11. Zhou, B., Li, X., Zang, T., Cai, Y. et al., "The Detection of False Data Injection Attack for Cyber–Physical Power Systems Considering a Multi-Attack Mode," *Applied Sciences* 13, no. 19 (2023), doi:https://doi.org/10.3390/app131910596.
12. Wang, J., Sargolzaei, A., Sargolzaei, S., Yen, K. et al., "Advanced Driver Assistance Systems: A Pathway to Autonomous Vehicles," *IEEE Access* 7 (2019): 107205-107226.
13. Zlomislić, V., Fertalj, K., and Sunk, V., "Denial of Service Attacks, Defences and Research Challenges," *Cluster Computing* 20 (2017): 661-671, doi:https://doi.org/10.1007/s10586-017-0730-x.
14. Bellardo, J. and Savage, S., "802.11 Denial-of-Service Attacks: Real Vulnerabilities and Practical Solutions," in *Proceedings of the 12th USENIX Security Symposium*, Washington, DC, 2003, 15-28, https://www.usenix.org/legacy/events/sec03/tech/full_papers/bellardo/bellardo.pdf.
15. Naha, A., Teixeira, A., Ahlen, A., and Dey, S., "Sequential Detection of Replay Attacks," *IEEE Transactions on Automatic Control* 68, no. 3 (2020): 1941-1948.
16. Jurcut, A.D., Coffey, T., and Dojen, R., "On the Prevention and Detection of Replay Attacks Using a Logic-Based Verification Tool," in *Computer Networks*, Kwiecień, A., Gaj, P., and Stera, P., Eds. (Cham: Springer International Publishing, 2014), 128-137.
17. Basit, A., Zafar, M., Liu, X., Javed, A.R. et al., "A Comprehensive Survey of AI-Enabled Phishing Attacks Detection Techniques," *Telecommunication Systems* 76, no. 1 (2021): 139-154, doi:https://doi.org/10.1007/s11235-020-00733-2.
18. Sun, Q., Miao, X., Guan, Z., Wang, J. et al., "Spoofing Attack Detection Using Machine Learning in Cross-Technology Communication," *Security and Communication Networks*

Downloaded from SAE International, Friday, January 31, 2025

2021 (2021): 3314595, doi:https://doi.org/10.1155/2021/3314595.

19. Sargolzaei, A., Crane, C.D., Abbaspour, A., and Noei, S., "A Machine Learning Approach for Fault Detection in Vehicular Cyber-Physical Systems," in *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, IEEE, Anaheim, CA, 2016, 636-640.
20. Noei, S., Sargolzaei, A., Abbaspour, A., and Yen, K., "A Decision Support System for Improving Resiliency of Cooperative Adaptive Cruise Control Systems," *Procedia Computer Science* 95 (2016): 489-496.
21. Alnaser, A., Akbas, M., Sargolzaei, A., and Rahul, R., "Autonomous Vehicles Scenario Testing Framework and Model of Computation," *SAE Intl. J CAV* 2, no. 4 (2019): 205-218, doi:https://doi.org/10.4271/12-02-04-0015.
22. Banerjee, S.S., Jha, S., Cyriac, J., Kalbarczyk, Z.T. et al., "Hands Off the Wheel in Autonomous Vehicles?: A Systems Perspective on Over a Million Miles of Field Data," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, Luxembourg, June 2018, 586-597.
23. Gietelink, O., Ploeg, J., Schutter, B.D., and Verhaegen, M., "Development of Advanced Driver Assistance Systems with Vehicle Hardware-in-the-Loop Simulations," *Vehicle System Dynamics* 44, no. 7 (2006): 569-590.
24. Bullock, D., Johnson, B., Wells, R., Kyte, M. et al., "Hardware-in-the-Loop Simulation," *Transportation Research Part C: Emerging Technologies* 12 (2004): 73-89.
25. Koopman, P. and Wagner, M., "Challenges in Autonomous Vehicle Testing and Validation," *SAE Int. J. Trans. Safety* 4, no. 1 (2016): 15-24, doi:https://doi.org/10.4271/2016-01-0128.
26. Mullins, G.E., Stankiewicz, P.G., Hawthorne, R.C., and Gupta, S.K., "Adaptive Generation of Challenging Scenarios for Testing and Evaluation of Autonomous Vehicles," *Journal of Systems and Software* 137 (2018): 197-215.
27. Chen, Y., Chen, S., Zhang, T., Zhang, S. et al., "Autonomous Vehicle Testing and Validation Platform: Integrated Simulation System with Hardware in the Loop," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, Changshu, China, 2018, 949-956.
28. Gambi, A., Mueller, M., and Fraser, G., "Automatically Testing Self-Driving Cars with Search-Based Procedural Content Generation," in *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*, ser. ISSTA 2019 (New York: Association for Computing Machinery, 2019), 318-328.
29. Li, N., Oyler, D., Zhang, M., Yildiz, Y. et al., "Game-Theoretic Modeling of Driver and Vehicle Interactions for Verification and Validation of Autonomous Vehicle Control Systems," arXiv:1608.08589 [cs], August 2016.
30. Kim, B., Kashiba, Y., Dai, S., and Shiraishi, S., "Testing Autonomous Vehicle Software in the Virtual Prototyping Environment," *IEEE Embedded Systems Letters* 9, no. 1 (2016): 5-8.
31. Zofka, M. René, S. Klemm, F. Kuhnt, T. et al., "Testing and Validating High Level Components for Automated Driving: Simulation Framework for Traffic Scenarios," in *IEEE Intelligent Vehicles Symposium (IV)*, Gothenburg, Sweden, 2016, 144-150.
32. Tuncali, C.E., Fainekos, G., Ito, H., and Kapinski, J., "Simulation-Based Adversarial Test Generation for Autonomous Vehicles with Machine Learning Components," in *IEEE Intelligent Vehicles Symposium (IV)*, Changshu, China, 2018, 1555-1562.
33. Jenkins, I.R., Gee, L.O., Knauss, A., Yin, H. et al., "Accident Scenario Generation with Recurrent Neural Networks," in *21st International Conference on Intelligent Transportation Systems (ITSC)*, Maui, HI, 2018, 3340-3345.
34. Tian, Y., Pei, K., Jana, S., and Ray, B., "Deep Test: Automated Testing of Deep-Neural-Network-Driven Autonomous Cars," in *Proceedings of the 40th International Conference on Software Engineering*, ACM, Gothenburg Sweden, 2018, 303-314.
35. Schultz, A., Grefenstette, J., and De Jong, K., "Adaptive Testing of Controllers for Autonomous Vehicles," in *Proceedings of the 1992 Symposium on Autonomous Underwater Vehicle Technology*, Washington, DC, 1992, 158-164.
36. Fremont, D.J., Kim, E., Pant, Y.V., Seshia, S.A. et al., "Formal Scenario-Based Testing of Autonomous Vehicles: From Simulation to the Real World," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, Rhodes, Greece, 2020, 1-8.
37. Ma, Y., Sun, C., Chen, J., Cao, D. et al., "Verification and Validation Methods for Decision-Making and Planning of Automated Vehicles: A Review," *IEEE Transactions on Intelligent Vehicles* 7, no. 3 (2022): 480-498.
38. Schwarz, C. and Wang, Z., "The Role of Digital Twins in Connected and Automated Vehicles," *IEEE Intelligent Transportation Systems Magazine* 14, no. 6 (2022): 41-51.
39. Donà, R., Vass, S., Mattas, K., Galassi, M.C. et al., "Virtual Testing in Automated Driving Systems Certification. A Longitudinal Dynamics Validation Example," *IEEE Access* 10 (2022): 47661-47672.
40. Feng, S., Sun, H., Yan, X., Zhu, H. et al., "Dense Reinforcement Learning for Safety Validation of Autonomous Vehicles," *Nature* 615, no. 7953 (2023): 620-627.
41. Chance, G., Ghobrial, A., McAreavey, K., Lemaignan, S. et al., "On Determinism of Game Engines Used for Simulation-Based Autonomous Vehicle Verification," *IEEE Transactions on Intelligent Transportation Systems* 23, no. 11 (2022): 20538-20552.
42. Alnaser, A.J., Sargolzaei, A., and Akbas, M.I., "Autonomous Vehicles Scenario Testing Framework and Model of Computation: On Generation and Coverage," *IEEE Access* 9 (2021): 60617-60628.
43. Watter, M., Springenberg, J.T. et al., "Learning a Predictive Model of the Environment for Model-Based Reinforcement Learning," arXiv preprint arXiv:1502.05361, 2015.
44. Bojarski, M. et al., "End-to-End Learning of Driving Models for Simulation and Control," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, Stockholm, Sweden, 2016.

Downloaded from SAE International, Friday, January 31, 2025

45. Perera, C. et al., "Generation of Virtual Scenarios for Testing Autonomous Vehicles Using Progressive Growing of GANS," in *IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, Ponta Delgada, Portugal, 2020.
46. Kim, J. et al., "Generating Multi-Modal Road Scenarios for Autonomous Driving Using GANS," arXiv preprint arXiv:2103.00020, 2021.
47. Hartmann, S., Weinmann,, M., Wessel, R., and Klein, R., "Streetgan: Towards road network synthesis with generative adversarial networks," (2017).
48. Shalev-Shwartz, S., Shammah, S., and Shashua, A., "On a Formal Model of Safe and Scalable Self-Driving Cars," arXiv preprint arXiv:1708.06374, 2017.
49. Noei, S., Parvizimosaed, M., and Noei, M., "Longitudinal Control for Connected and Automated Vehicles in Contested Environments," *Electronics* 10, no. 16 (2021): 1994.
50. Sargolzaei, A., Zegers, F., Abbaspour, A., Crane, C. et al., "Secure Control Design for Networked Control Systems with Nonlinear Dynamics under Time-Delay-Switch Attacks," *IEEE Transactions on Automatic Control* 68, no. 2 (2022): 798-811.
51. Sargolzaei, A., Allen, B.C., Crane, C.D., and Dixon, W.E., "Lyapunov-Based Control of a Nonlinear Multiagent System with a Time-Varying Input Delay under False-Data-Injection Attacks," *IEEE Transactions on Industrial Informatics* 18, no. 4 (2021): 2693-2703.
52. Sargolzaei, A., "A Secure Control Design for Networked Control System with Nonlinear Dynamics under False-Data-Injection Attacks," in *2021 American Control Conference (ACC)*, IEEE, New Orleans, LA, 2021, 2693-2699.
53. Wang, Y., Wang, Y., Qin, H., Ji, H. et al., "A Systematic Risk Assessment Framework of Automotive Cybersecurity," *Automotive Innovation* 4, no. 3 (2021): 253-261, doi:https://doi.org/10.1007/s42154-021-00140-6.

© 2024 The Authors. Published by SAE International. This Open Access article is published under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0/), which permits distribution, and reproduction in any medium, provided that the original author(s) and the source are credited.

Positions and opinions advanced in this work are those of the author(s) and not necessarily those of SAE International. Responsibility for the content of the work lies solely with the author(s).

Downloaded from SAE International, Friday, January 31, 2025